Claims

What is claimed is:

1. A method for storing a data block, comprising:
   storing the data block in a storage pool;
   obtaining a data block location;
   determining a checksum function for the data block;
   calculating a data block checksum using the checksum function for the data block; and
   storing a first indirect block in the storage pool, wherein the first indirect block comprises the data block location, the data block checksum, and a checksum function ID corresponding to the checksum function for the data block.

2. The method of claim 1, further comprising:
   determining a checksum function for a first indirect block, wherein the checksum function for the first indirect block is associated with a checksum function ID corresponding to the checksum function for the first indirect block;
   calculating a first indirect block checksum using the checksum function for a first indirect block;
   obtaining a first indirect block location; and
   storing a second indirect block in the storage pool, wherein the second indirect block comprises the first indirect block location, the first data block checksum, and checksum function ID corresponding to the checksum function for the first indirect block.

3. The method of claim 1, further comprising:
   assembling the first indirect block, wherein assembling the first indirect block comprises populating a block pointer in the first indirect block.

4. The method of claim 3, wherein populating the block pointer comprises:
   storing the data block checksum in a checksum field in the block pointer,

storing the checksum function ID for the data block in a checksum function ID field in the block pointer; and

storing the data block location in the block pointer, wherein storing the data block location comprises storing a metaslab ID and offset.

5. The method of claim 4, further comprising:

storing a birth value in a birth field in the block pointer.

6. The method of claim 3, wherein the first indirect block is assembled using a data management unit.

7. The method of claim 1, wherein the storage pool comprises at least one storage device.

8. The method of claim 1, wherein the storage pool is divided into a plurality of metaslabs.

9. The method of claim 8, wherein each of the plurality of metaslabs is associated with a metaslab ID.

10. The method of claim 8, wherein the data block location comprises the metaslab ID and an offset.

11. The method of claim 1, wherein storing the data block comprises using a storage pool allocator.

12. A method for storing a first data block and a second data block, comprising:

storing the first data block and the second data block in a storage pool;

obtaining a first data block location and a second data block location;

calculating a first data block checksum for the first data block using a first checksum function, wherein the first checksum function is associated with a first checksum ID;

calculating a second data block checksum for the second data block using a second checksum function, wherein the second checksum function is associated with a second checksum ID; and

storing an array of block pointers in an indirect block, wherein the array of block pointers comprises:

a first block pointer comprising the first data block location, the first data block checksum, and the first checksum ID, and

a second block pointer comprising the second data block location, the second data block checksum, and the second checksum ID.

13. The method of claim 12, wherein the indirect block is assembled using a data management unit.

14. The method of claim 12, wherein the indirect block is stored using a storage pool allocator.

15. A method for retrieving data in a data block, comprising:

obtaining an indirect block comprising a stored checksum, a stored checksum function ID, and a data block location;

obtaining the data block using the data block location;

calculating the checksum for the data block using a checksum function corresponding to the stored checksum ID to obtain a calculated checksum;

retrieving the data from the data block, if the stored checksum equals the calculated checksum; and

performing an appropriate action, if the stored checksum is not equal to the calculated checksum.

16. The method of claim 15, wherein the indirect block is assembled using a data management unit

17. The method of claim 15, wherein the calculated checksum is calculated using a storage pool allocator.

18. A method for storing and retrieving a data block, comprising:

storing the data block in a storage pool;

obtaining a data block location;

determining a checksum function for the data block;

calculating a data block checksum using the checksum function for the data block;

storing an indirect block in the storage pool, wherein the indirect block comprises the data block location, the data block checksum, and a checksum function ID corresponding to the checksum function for the data block;

obtaining the indirect block comprising the data block checksum, the checksum function ID, and the data block location;

obtaining the data block using the data block location;

calculating the checksum for the data block using a checksum function corresponding to the checksum ID to obtain a calculated checksum;

retrieving the data from the data block, if the data block checksum equals the calculated checksum; and

performing an appropriate action, if the data block checksum is not equal to the calculated checksum.

19. A system for storing a data block, comprising:

a storage pool comprising the data block and the first indirect block, wherein the first indirect block comprises a data block checksum, a data block checksum function ID, and a data block location; and

a storage pool allocator configured to store:

the data block and the first indirect block in the storage pool, and

a plurality of checksum functions, wherein each of the plurality of checksum functions is associated with a checksum function ID.

20. The system of claim 19, further comprising:

a second indirect block, comprising a first indirect data block checksum, a first indirect data block checksum function ID, and a first indirect block location,

wherein the storage pool allocator is further configured to store the second indirect block in the storage pool.

21. The system of claim of claim 19, further comprising:

a data management unit configured to assemble the first indirect block and request the storage pool allocator store the first indirect block.

22. The system of claim 19, wherein the storage pool comprises at least one storage disk.

23. The system of claim 19, wherein the storage pool is divided into a plurality of metaslabs.

24. The system of claim 23, wherein each of the plurality of metaslabs is associated with a metaslab ID.

25. The system of claim 24, wherein the data block location comprises the metaslab ID and an offset.

26. A computer system for storing a data block, comprising:

a processor;

a memory;

a storage device; and

software instructions stored in the memory for enabling the computer system under control of the processor, to:

store the data block in a storage pool;

obtain a data block location;

determine a checksum function for the data block;

calculate a data block checksum using the checksum function for the data block; and

store a first indirect block in the storage pool, wherein the first indirect block comprises the data block location, the data block checksum, and a checksum function ID corresponding to the checksum function for the data block.

27. A network system having a plurality of nodes, comprising:

a storage pool comprising the data block and the first indirect block, wherein the first indirect block comprises a data block checksum, a data block checksum function ID, and a data block location; and

a storage pool allocator configured to store:

the data block and the first indirect block in the storage pool, and

a plurality of checksum functions, wherein each of the plurality of checksum functions is associated with a checksum function ID,

wherein the storage pool is located on any one of the plurality of nodes, and

wherein the storage pool allocator is located on any one of the plurality of nodes.